On the evolution of the SAM framework

Luca Magnoni
DRAFT v.0.4
7 March 2014

SAM in pills

- Monitoring functionality of Grid services:
  - Schedule checks via custom probes for:
    - Public Grid services (FTS, LFC, SRM, Proxy, ...)
    - Job-submission
    - WN tests (executed on remote WN), submitted as job payload
  - Message broker as transport layer
    - Forward checks results (with local cache)
    - Gather checks results

    - Metrics visualization
    - Ability to attach handlers to checks (e.g. low level notifications)
    - Compute (and re-calculate) site's reliability and availability
- Nagios based
- Gridmon utility provided to simplify probe development
    - Heavily used in job submission and storage probes (for test dependency)
- NCG as Nagios Configuration Generator
    - Get topology and profiles from API
    - Per-experiment configuration via static files/RPMs
    - No templates
- Some numbers:
    - ~ 30 probes
        - from experiments, SAM and Product Teams
    - ~ 113 metrics
    - ~ 400k checks/day (~250k checks/day for ATLAS)

SAM maintenance - what needs to be done to run tests after 2014

- Move away from custom Nagios (5 patches pending)
    - Push patches mainstream and use standard Nagios ?
    - Use different submission system?
- Lightweight messaging daemon (mta-simple) is obsolete
    - Used both in local SAM and WN for results forwarding
    - MIG provide new full MTA toolchain (dirq, MTA daemon, simplevisor). Seems appropriate for SAM machines, not for remote WNs
- Support for UMD3 and SL6 with continuous integration tests
- Move to puppet
- Gridmon framework revamp
    - CREAM and Condor-G probes strong dependency
    - Gridmon is Nagios specific
- Job submission probes code refactoring
    - possibly 1 JobMonit per endpoint
- Converge sites/experiment discussion on scheduling timeout
    - considering new direct CREAM and Condor-G based job submission

SAM Evolution

Requirements (from [TEG report] and [WLCG consolidation survey])

- Simplification
    - Simplify check management and configuration
        - Profit from centralize architecture

    - Templates, to decouple SAM configuration from submission technology (e.g. Nagios)
    - Less effort to run and maintain the system
- Improve WN checks
    - Lightweight solution
        - In some condition, Nagios on WN as impact on the memory footprint for the job
    - But still providing an execution framework users can rely on to develop probes
        - which handles probe execution on WN, timeouts, sending results to message broker, etc.
- Better scalability
    - Handle higher test frequency
    - Handle new type of test/probes (e.g xrootd, http)
    - Testing from multiple nodes (test partitioning)
    - Testing of services not in GOCDB (testing any services)
- Improvements
    - Configure checks with finer granularity than service endpoint (e.g. queue name, space token, FQANs)
    - Improve efficiency of the job submission (e.g. with multiple FQANs, space tokens, etc)
    - Ability to re-run checks on demand
    - Service-level credentials management (e.g. proxy for T1 sites only)
    - More advanced check dependency ?
        - may be needed for probes not using the Gridmon framework
- Better availability calculation
    - On demand?
    - Proper interface to change result?

SAM conceptual model

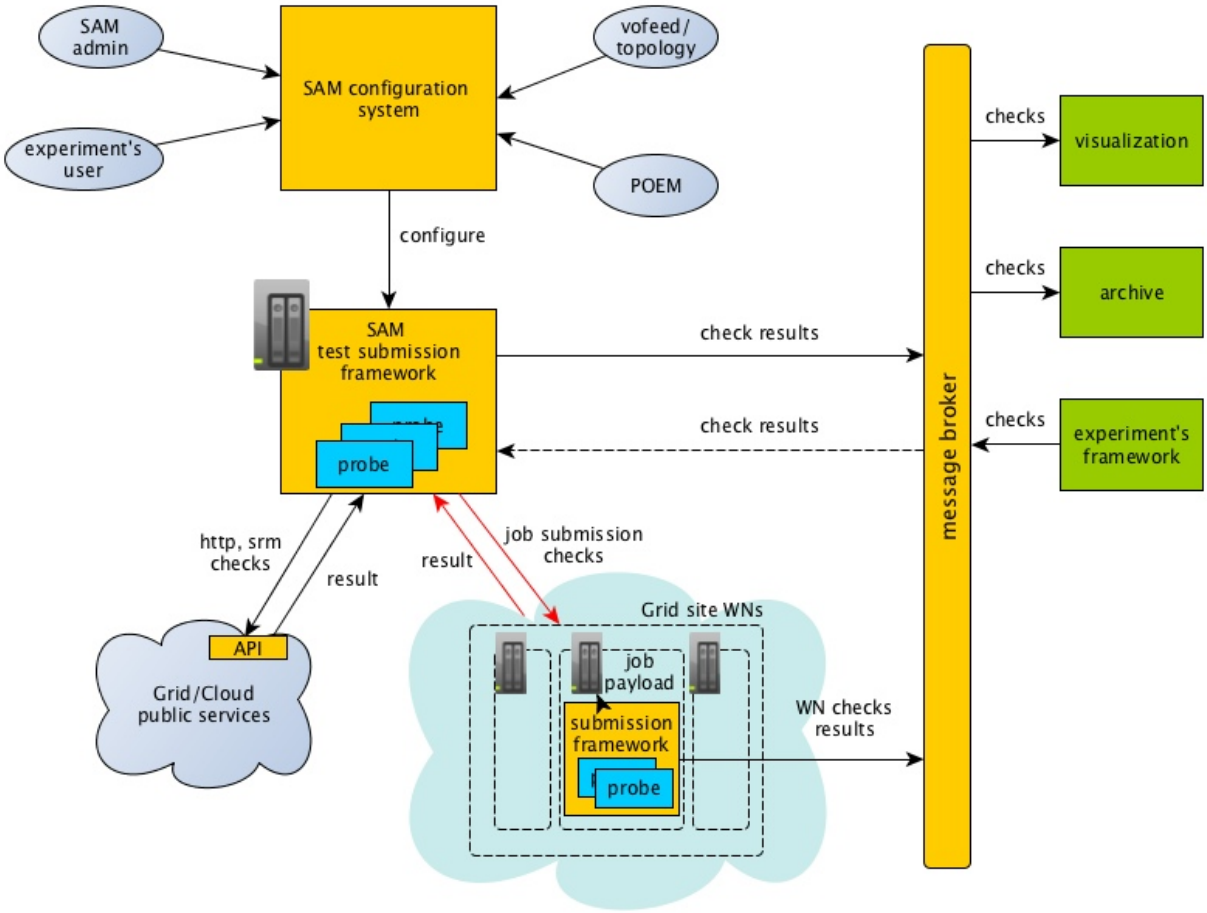

Fig 1: SAM conceptual architecture

SAM today

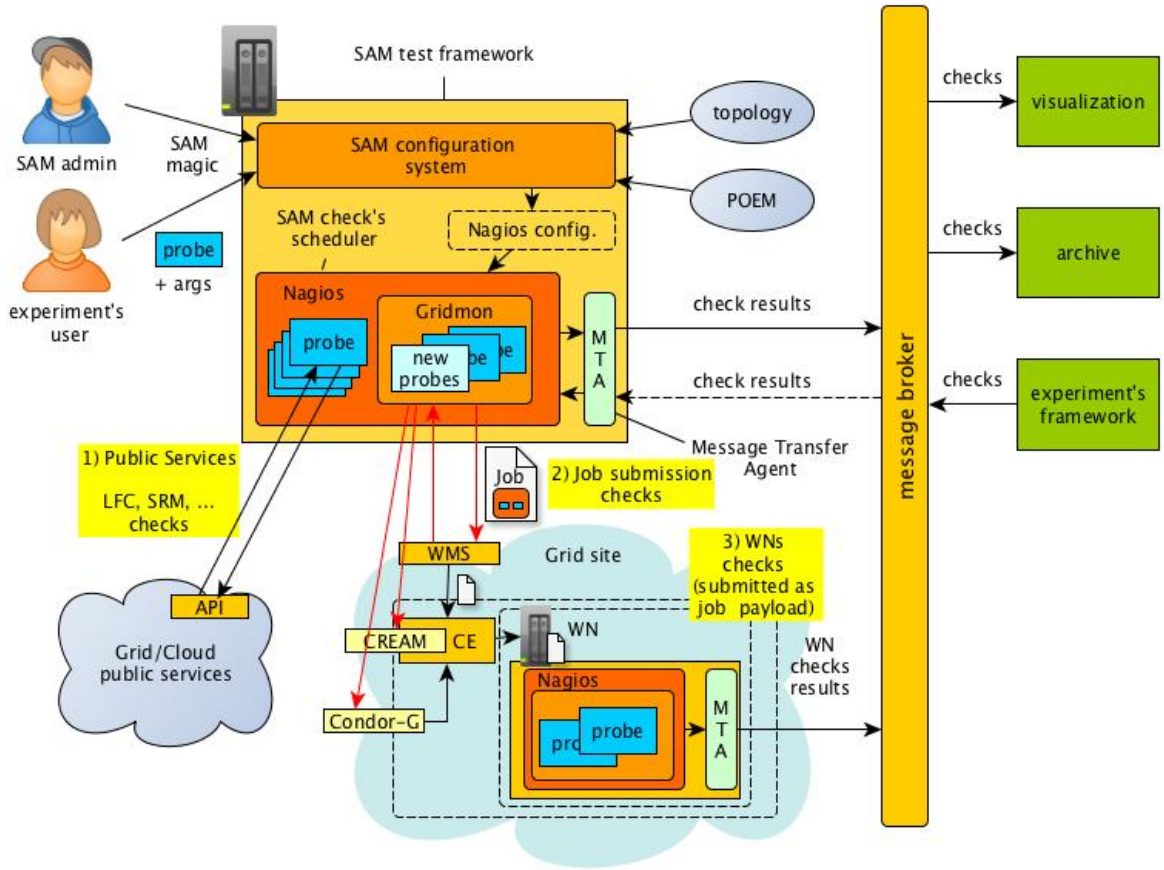

Fig 2: Current SAM components - high-level view

Areas of evolution

1) Configuration

Today, SAM check's configuration is done via RPMs & static files. The Nagios Configuration Generator (NCG) utility parses those configuration files and generates the final Nagios configuration. Experiments have to build their own rpm with probes configuration and provide them to SAM admin. The system can evolve toward web-based configuration for checks and probes, and lighter NCG to read config information via REST API and build actual Nagios file.

Concepts

- probe options: path to executable, list of supported arguments
- scheduling options: how often check is executed, dependency, timeout, retry, etc.
- metrics = probe + arguments+ scheduling
- poem: bind metrics (by name) to service flavour
- topology: bind endpoint to service flavour

For a real example of options, see the Appendix.

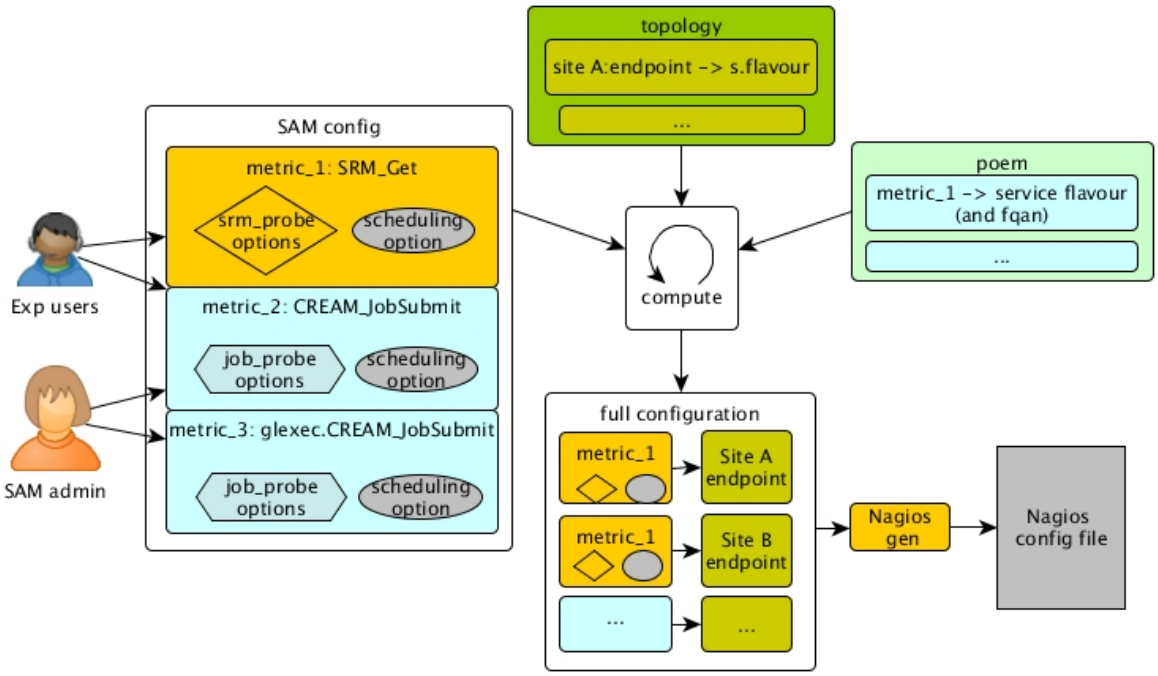

Fig 3: Concepts and configuration workflow for SAM

Limitation of the current configuration system

- Not able to configure probes with finer granularity than service endpoint (e.g. queue names, space token, FQANs)
- No service-level credential management
- No templates, strictly coupled with Nagios
- Not so good experiments experience
    - users provide rpm for configuration files
    - users not able to check final configuration (build only by NCG)
- Not easy to scale
    - e.g. multiple production machines -> requires configuration partitioning
- SAM admin deeply involved
- Relational database is needed locally

Proposal

- Introduce a central point of configuration
    - DB based
    - Web API and UI
        - Experiment entry point to configure/add probe parameters
    - Read topology and metrics to build full view for users
- With a lightweight configuration generator that fetches from central config and generates puppet/Nagios config (with templates for different submission systems)
    - similarities with MIG MBSD approach

2) Simplify WN check framework

A check framework is needed to handle the execution of checks on the remote WN machines. It handles configuration, probe execution, timeouts, check termination, sandboxing and result propagation. Reliability is crucial in order to guarantee that tests has only negligible impact on WN resources. Currently, a stripped-down Nagios version is shipped together with probes as Job payload.

Nagios on WN

Limitations:

- Memory footprint
- Complexity

Pros:

- Handles checks execution
    - timeouts, retry
- Creates sandbox for checks
- Provide checks configuration environment
    - Nagios compliant
    - input/output structure
- Forward check results to directory queue

Investigation on alternative WN framework

collectd [see Appendix] can be considered as an alternative lightweight submission framework for the WN. It has a Nagios plugin to execute Nagios-compliant probes with a modular architecture for results forwarding. Nevertheless, is meant to collect metrics rather than run checks so it can have limitations, in particular for passive metrics.
MIG-team is considering collectd for the new monitoring project, in case profit from common work.

3) Improve SAM check framework

Nagios is at the core of SAM as the check scheduler which runs active active tests on remote sites and services.

Nagios in pills

- Results are produced by Active (plugins) / Passive (result from file) checks
    - in SAM, plugins are called “probes”
- Plugins are executable (command line) scripts/executable compliant to a simple API (0-4 return code, stdout output)
    - actually, Nagios API is more then that and define performance data as well as input arguments...
- Plugins are executed in parallel

- Hosts are (usually) physical devices with one or more services assigned, one address (map, ipv4, ipv6), can have parents relation in a topology layout
- Services can be host attributes (CPU load, disk, etc), services hosted (http, ftp, etc.), other things (DNS records)
- Service state mapped to plugin result (OK, WARNING, CRITICAL, UNKNOWN)
- Host state (UP/DOWN/UNREACHABLE) (re)computed on service status changes, with check on host parents to declare un-reachability

Nagios features used in SAM:

- Schedule checks
    - on regular interval
        - computed by the last check result
    - handle timeouts
        - execution timeouts
        - result validity (e.g. result to UNKNOWN/MISSING)
    - handle retries
        - interval and max number of attempts
- Handles check/service states
    - detect flipping
- Define check dependency
    - conditional checks (e.g. if A=ERROR check B, if C=OK check D, etc.)
        - e.g. myproxy dependency
- Gather passive result
    - this allow probes, as current SRM, to have several checks executed in order (via Gridmon) and report multiple results
- Sandboxing check
    - checks executed in parallel
- Manage message bus integration
    - Forward check results
    - Receive passive from bus
- Manage credentials
- Handle notifications
- Web UI to visualize check submission information, results, stdout, etc.
    - re-execute check, only for active

While being a solid and reliable solution, Nagios suffers from limitations in particular on the overall system scalability and flexibility.

Nagios alternatives

In recent years, several tools appeared as Nagios alternatives. While they all aim at overcoming Nagios limitation, they took different approaches. Project as Icinga, Zabbix or Shinken can be considered as Nagios rewrite, with a static configuration for services and

checks, but more advanced UI and analysis capability. Others, like Zenoss and Sensu, moved toward a decentralized configuration for services and checks.

- @todo best candidate analysis

4) Towards metrics inference and on-demand testing

The first 3 areas of evolution aim at improving user interface and background technologies, but the overall architecture is not affected. Nevertheless, the current system which is based on remote periodic checks execution to estimate service and site statuses has shown clear limitations (or better, trade-offs). For example, a site operating at full speed for production jobs can be wrongly reported as unavailable because there is no room for the check to run.

Metrics inference and conditional testing

A possible evolution is to infer the status of sites and services from the monitoring data reported by experiment frameworks and other monitoring tools, and execute test to complement the computed data if needed. This will require:

- A processing engine able to analyze the streams of monitoring data and to infer the status of a service, processing monitoring information and logs. It should also detect the condition which may trigger a check request to complement the metric generation (e.g. suspicious error on data transfer).
- A check framework able to receive commands on-demand. Nagios can do the job but other framework as Zabbix can do it better.
- A transport layer to distribute commands and check results. Messaging seems appropriate.

Conservative strategy

After maintenance, start with configuration improvement and simplification (1), this will simplify SAM-user's experience and admin's work, preserving the current system architecture while decoupling SAM test configuration from underlying scheduler (Nagios). On the WN tests (2) , configuration of probes and payload can be improved but no clear alternative to Nagios as WN test scheduler has been identified so far. The same is true for the main SAM check scheduler (3), Nagios alternative exists and they solve known Nagios limitations, but for SAM use case they could only bring little value while demanding integration and development effort. In case, a deeper analysis should be done on a small set of candidate tools. Work on metrics inference and on-demand testing (4) can be done independently on other the SAM components, but it has implication on the check scheduler capability when it comes to run checks on demand.

Appendix

More on SAM job submission

[https://indico.cern.ch/getFile.py/access?contribId=0&resId=0&materialId=slides&confId=283876]

Technology Evaluation

collectd

collectd is a daemon which collects system performance and other statistics periodically and provides mechanisms to store the values in a variety of ways, for example in RRD files, or CSV files, or AMQP.
Small footprint, simple configuration, chef/puppet ready. Deeply plug-in based architecture, both for getting metrics as for writing result. Plugin can be developed in C/Perl/Java/Python/executable bash script.
4 data type: gauges, derive , counter, absolute
Nice presentation at last monitoriama eu:
https://speakerdeck.com/monitorama/berlin-2013-collectd-workshop-florian-forster
Possible weak point: strict naming schema host+plugin_type
Additional good point: probably used in MIG monitoring evolution, so collectd2broker should come for free